HYLO: Hibridni logaritmični množilnik za energijsko učinkovito računanje

Ratko Pilipović, Patricio Bulić
University of Ljubljana, Faculty of Computer and Information Science, Večna pot 113, 1000 Ljubljana, Slovenia
ratko.pilipovic@fri.uni-lj.si, patricio.bulic@fri.uni-lj.si

Izvleček

V članku predstavimo hibridni logaritmični množilnik, HYLO, ki združuje Radix-4 Boothovo kodiranje in logaritmično aproksimacijo za generiranje delnih produktov. HYLO tako prinaša prednosti dveh glavnih strategij v domeni približnih množilnikov, rezanje delnih produktov in logaritemsko aproksimacijo množenja. Zaradi manjšega števila delnih produktov, HYLO množilnik ima manjšo porabo toka, zato ga lahko uporabimo v široki paleti aplikacij, ki so imune na računsko napako, brez degradacije natančnosti ali performans.

Ključne besede: Logaritmični množilnik, načrtovanje aritmtičnih vezj, približni množilniki, približno računanje

Abstract

We propose a novel hybrid logarithmic approximate multiplier HYLO, which combines Radix-4 Booth encoding and logarithm product approximation for partial product generation. With this, HYLO brings the advantages of two major strategies: truncated multipliers and logarithm product approximation. Due to the smaller number of partial products, the described multiplier is power-efficient and can be employed in a wide area of error-resilient applications without the decrease in performance or quality.

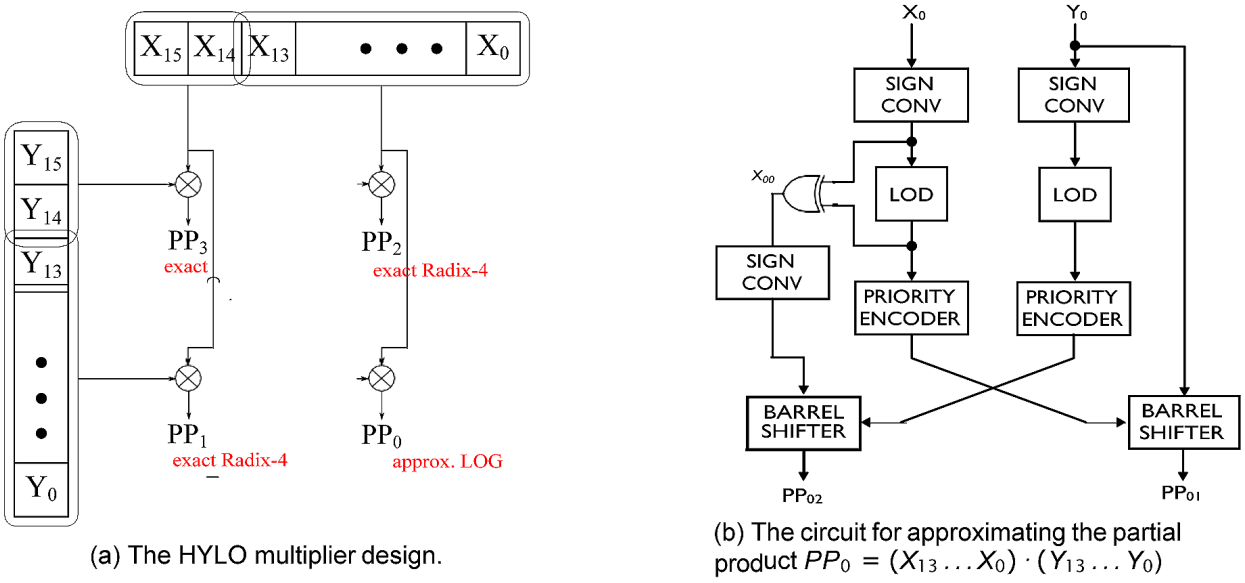
Keywords: Approximate computing, approximate multiplier, arithmetic circuit design, logarithmic multipliers

1 INTRODUCTION

The explosion of the volume of complex and noisy data [Agrawal et al., 2016], produced by low-power devices and scientific data centres, stretches the capabilities of modern computing platforms and raises concern on power-efficient processing in embedded systems and high-performance computing platforms. Due to the emergence of error-tolerant applications [Jerger and Miguel, 2018], approximate computing surfaced as the solution for achieving power-efficient processing [Mittal, 2016]. Approximate computing allows small inaccuracies in computing in order to achieve more efficient processing. Recent progress in the area of approximate computing [Jerger and Miguel, 2018, Eeckhout, 2018] indicates the popularity of this approach.

Traditional arithmetic circuits are encountering difficulties when it comes to design improvement [Azizi et al., 2010]. This made the approximate computing a popular strategy for arithmetic circuit design [Han and Orshansky, 2013] where energy-efficient design is more important than accuracy and the challenge is to achieve the best trade-off between accuracy and design efficiency. Multipliers represent complex arithmetic circuits [Parhami, 1999] and at the same time are indispensable components for many systems. Considering this, more efficient design of a multiplier is needed for more energy-efficient processing. With approximate computing techniques, we can achieve efficient multiplier design and improve the performance of error-resilient systems.

Two design paradigms dominate in the field of approximate multipliers: logarithmic and truncated multipliers. Logarithmic multipliers rely on the addition of binary logarithms to approximate multiplication. They employ the property that multiplication can be substituted with addition in the logarithmic domain. Mitchell's multiplier [Mitchell, 1962] represents the first logarithmic multiplier, where the binary logarithm is used to approximate multiplication. In order to improve Mitchell's multiplier design, Babic et al. developed the iterative logarithmic multiplier (ILM) [Babić et al., 2011], that achieves arbitrary accuracy through an iterative procedure.



(a) The HYLO multiplier design.

(b) The circuit for approximating the partial product $PP_0 = (X_{13} \ldots X_0) \cdot (Y_{13} \ldots Y_0)$

Figure 1: **16-bit HYLO multiplier. (a) The proposed multiplier generates four partial products, PP_3, PP_2, PP_1, and PP_0. The partial products PP_3, PP_2 and PP_1 are exact and simple to generate using the Radix-4 Booth encoding. (b) The partial product PP_0 is generated using the logarithmic multiplier approximation strategy.**

Recently, the number rounding approach became a popular approach for logarithmic multiplier design. Zendagani et al. [Zendegani et al., 2017] developed the rounding based approximate (ROBA) multiplier, where the input operands are rounded to the nearest value of the power of two.

The error of the truncated multipliers emerges from simplifying two main stages of a multiplier: the partial product addition and the partial product generation. Simplifications in the partial product addition stage rely on employment of approximate compressors. [Yang et al., 2015]. In order to deliver efficient compressor, Esposito et al. [Esposito et al., 2018] developed XOR-free compressors that achieve smaller error and area usage. Simplifications in PP generation could lead to more efficient design than simplifications in the partial product addition stage. The goal is to produce less partial products and at the same time produce a small error. Following this, Zervakis et al. [Zervakis et al., 2015] introduced the technique for the partial product omission in Booth encoding. The omission of partial products leads to the simpler partial products addition stage but leads to a significant decrease in accuracy.

Previous work in the field of approximate multipliers indicates its importance and necessity for power efficient processing. We can also see that there is no clear winner in this field and that the search for an efficient approximate multiplier is far from being finished. In this work, we introduce the Hybrid Logarithm (HYLO) multiplier which utilizes the concept of operand decomposition to generate fewer partial products compared to the exact multiplier. HYLO incorporates Radix-4 Booth encoding to encode two partial products from the two most significant bits and a logarithmic multiplier to generate one partial product from the other bits. The proposed multiplier combines advantages from both worlds: simple design from logarithmic multipliers and high accuracy of truncated multipliers.

The rest of paper is organized as follows: Section 2 describes the HYLO product approximation and Section 3 present error and design evaluation of HYLO multiplier. Finally, The Section 4 concludes thepaper.

2 THE PROPOSED HYLO MULTIPLIER

HYLO multiplier addresses two important issues. The first issue refers to the design of the efficient partial product (PP) generation stage. By combining Radix-4 Booth encoding [Ercegovac and Lang, 2004] and logarithmic multiplier approximation strategy, HYLO tries to deliver good trade-off between low-power design and small error. The second issue refers to the challenge of increasing the accuracy of logarithmic multiplier without a significant increase in area utilization and power consumption. HYLO multiplier uti-

Table 1: **Synthesis results and MRE.**

Multiplier	Delay (ns)	Power (*mW*)	Area (μm_2)	Energy (*pJ*)	PADP ($mJ \cdot \mu m_2$)	MRE (%)
Radix-4	4.05	1.526	6.23e+04	6.18	0.385	0.0
HYLO	6.43	0.796	5.56e+04	5.11	0.261	4.13

lizes the concept of operand decomposition to address previously mentioned problems. Illustration of HYLO multiplier design is presented in Figure 1a.

Let X and Y be the 16-bit signed numbers in two's complement representation. To obtain the product $X \cdot Y$, both inputs X and Y are divided into two segments: most significant segments (MSS) - X_1, Y_1 and least significant segments (LSS) - X_0, Y_0. The MSS consists of two leftmost bits while other bits belong to LSS. Then, segments are multiplied with each other and added at the end:

$$X \cdot Y = X_1 \cdot Y_1 \cdot 2^{28} + (X_1 \cdot Y_0 + X_0 \cdot Y_1) \cdot 2^{14} + X_0 \cdot Y_0 \qquad (1)$$

The circuit for generation of partial product $X1$ $Y1$ is implemented through employment of 2-bit signed multiplier. For designing 2-bit signed multiplier we employed Carnaugh's minimization procedure. Partial products $X_1 \cdot Y_0$ and $Y_1 \cdot X_0$ are obtained through employment of Radix-4 Booth encoding:

$$X_1 \cdot Y_0 = \hat{x}^{R4} \cdot Y_0 \quad \text{and} \quad Y_1 \cdot X_0 = \hat{y}^{R4} \cdot X_0 \qquad (2)$$

where:

$$x^{R4} = -2 \cdot x_{15} + x_{14} + x_{13} \quad \text{and} \quad y^{R4} = -2 \cdot y_{15} + y_{14} + y_{13} \qquad (3)$$

The partial products X_1 Y_0 and Y_1 Y_0 are easy to generate without multiplication due to the use of radix-4 Booth encoding. The only multiplication in Eq. (1) is required by the term X_0 Y_0. Note that X_0 and Y_0 are 14-bit signed integer numbers. According to study [Babić et al., 2011], we approximate partial product $X_0 \cdot Y_0$ as:

$$X_0 \cdot Y_0 \approx sign(X_0) Y_0 2^{k_{X_0}} + sign(Y_0) X_{00} 2^{k_{Y_0}} = PP_{01} + PP_{02} \qquad (4)$$

where $k_{Y_0} = \lfloor log_2 | Y_0 | \rfloor$, $k_{X_0} = \lfloor log_2 | X_0 | \rfloor$, $X_{00} = | X_0 | - 2^{k_{X_0}}$ and $sign(X_0 \cdot Y_0)$ denotes the sign of product $X_0 \cdot Y_0$.

The circuit for partial product generation is shown in Figure 1b. Sign conversion blocks (SIGN CONV) are employed to calculate the absolute value of input operands and sign of partial products.Next, Leading-one-detectors (LOD) together with Priority encoders calculate the k_X0 and k_Y0 . In the end, Barell shifters produce partial products. Finally, all partial products are added by employing the Wallace tree methodology [Wallace, 1964]. For the final addition of partial products, we employ Carry-Look-Ahead adder [Ercegovac and Lang, 2004].

3 RESULTS AND DISCUSSION

This section presents the evaluation of proposed multiplier in terms of hardware (power, area, delay, energy and power-area-delay product (PADP)) and mean relative error (MRE). We have compared HYLO with an accurate radix-4 multiplier. The multipliers have been implemented in Verilog and synthesized using TSMC 180-nm standard cell library. After place-and--route, the cell area and critical path delay were reported. The power consumption is estimated after the layout extraction using the IRSIM simulator with randomly generated 10000 input test vectors. Table 1 contains the synthesis results (delay, power, area, energy and power-area-delay product) as well as MRE for both multipliers.

From Table 1 we can see that HYLO offers almost 50 % savings in power, and 10 % savings in chip area, but it has almost 50 % larger critical path delay. Nevertheless, in delivers 17 % savings in energy and almost 33 % in power-area-delay product (PADP), while keeping small MRE.

4 CONCLUSION

In this paper, we propose a design strategy for approximate multipliers, which combines the advantages of truncated and logarithmic multipliers. On the top level, the architecture of HYLO multiplier resembles the design of truncated multiplier. It employs operand decomposition and partial product addition to generate the product. On the other hand,

HYLO approximates partial product $X_0 \cdot Y_0$ using binary logarithms. This makes it similar to logarithmic multiplier. Because of this, HYLO represents a hybrid multiplier, which combines elements from truncated and logarithmic multipliers. This allows it to achieve a good trade-off between accuracy and design efficiency.

REFERENCES

[1] [Agrawal et al., 2016] Agrawal, A., Choi, J., Gopalakrishnan, K., Gupta, S., Nair, R., Oh, J., Prener, D. A., Shukla, S., Srinivasan, V., and Sura, Z. (2016). Approximate computing: Challenges and opportunities. In *2016 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–8.

[2] [Azizi et al., 2010] Azizi, O., Mahesri, A., Lee, B. C., Patel, S. J., and Horowitz, M. (2010). Energy-performance tradeoffs in processor architecture and circuit design: A marginal cost analysis. *SIGARCH Comput. Archit. News*, 38(3):26–36.

[3] [Babić et al., 2011] Babić, Z., Avramović, A., and Bulić, P. (2011). An iterative logarithmic multiplier. *Microprocessors and Microsystems*, 35(1):23–33.

[4] [Eeckhout, 2018] Eeckhout, L. (2018). Approximate computing, intelligent computing. *IEEE Micro*, 38(4):6–7. [Ercegovac and Lang, 2004] Ercegovac, M. D. and Lang, T. (2004). *Digital arithmetic*. Elsevier.

[5] [Esposito et al., 2018] Esposito, D., Strollo, A. G. M., Napoli, E., Caro, D. D., and Petra, N. (2018). Approximate multipliers based on new approximate compressors. *IEEE Transactions on Circuits and Systems I: Regular Papers*, pages 1–14.

[6] [Han and Orshansky, 2013] Han, J. and Orshansky, M. (2013). Approximate computing: An emerging paradigm for energy--efficient design. In *Test Symposium (ETS), 2013 18th IEEE European*, pages 1–6. IEEE.

[7] [Jerger and Miguel, 2018] Jerger, N. E. and Miguel, J. S. (2018). Approximate computing. *IEEE Micro*, 38(4):8–10.

[8] [Mitchell, 1962] Mitchell, J. N. (1962). Computer multiplication and division using binary logarithms. *IRE Transactions on Electronic Computers*, (4):512–517.

[9] [Mittal, 2016] Mittal, S. (2016). A survey of techniques for approximate computing. *ACM Computing Surveys (CSUR)*, 48(4):62.

[10] [Parhami, 1999] Parhami, B. (1999). *Computer arithmetic*, volume 20. Oxford university press.

[11] [Wallace, 1964] Wallace, C. S. (1964). A suggestion for a fast multiplier. *IEEE Transactions on electronic Computers*, (1):14–17.

[12] [Yang et al., 2015] Yang, Z., Han, J., and Lombardi, F. (2015). Approximate compressors for error-resilient multiplier design. In *2015 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, pages 183–186. IEEE.

[13] [Zendegani et al., 2017] Zendegani, R., Kamal, M., Bahadori, M., Afzali-Kusha, A., and Pedram, M. (2017). Roba multiplier: A rounding-based approximate multiplier for high-speed yet energy-efficient digital signal processing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(2):393–401.

[14] [Zervakis et al., 2015] Zervakis, G., Xydis, S., Tsoumanis, K., Soudris, D., and Pekmestzi, K. (2015). Hybrid approximate multiplier architectures for improved power-accuracy trade--offs. In *Low Power Electronics and Design (ISLPED), 2015 IEEE/ACM International Symposium on*, pages 79–84. IEEE.

[15] *Ratko Pilipović* received his B.Sc. and M.Sc. degrees from the Faculty of Electrical Engineering, University in Banjaluka, Bosnia and Hercegovina in 2015 and 2017, respectively. He is currently working towards the Ph.D. degree at the Faculty of Computer and Information Science, University of Ljubljana, Slovenia. His research interests include approximate computing, arithmetic circuit design, FPGA design, embedded processing and machine vision.

[16] *Patricio Bulić* received his B.Sc. degree in electrical engineering, and M.Sc. and Ph.D. degrees in computer science from the University of Ljubljana, Slovenia, in 1998, 2001 and 2004, respectively. He is an Associate Professor at the Faculty of Computer and Information Science, University of Ljubljana. His main research interests include computer architecture, digital design, approximate computing, computer arithmetics and parallel processing.